CS354: Machine Organization and Programming

Lecture 22
Friday the October 23rd 2015

Section 2
Instructor: Leo Arulraj

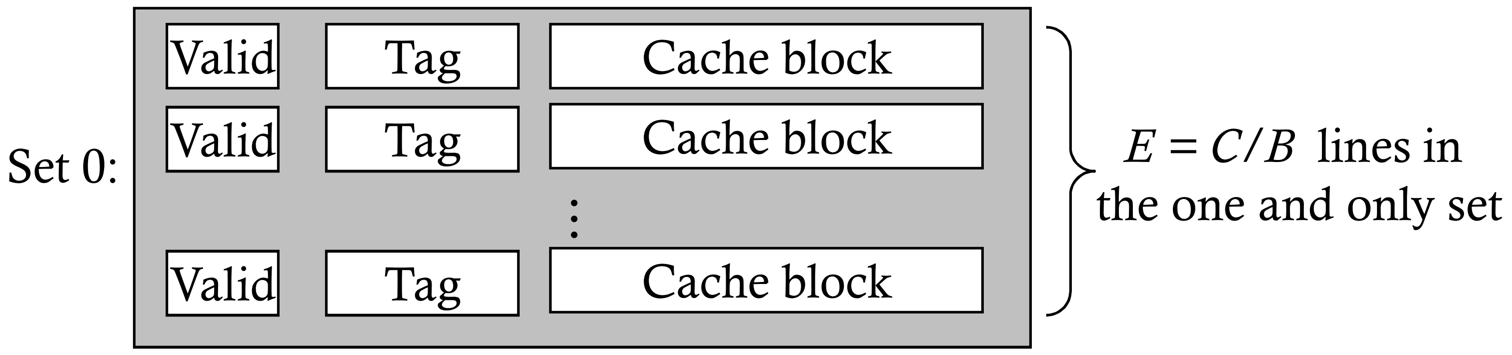
Class Announcements

1. Programming Assignment 3 released.
   Due by Nov 4th before 9 AM. Start early!
   Theme: Measurements and analysis of caches

Lecture Overview

1. Fully Associative caches
2. Write Policies
3. I-cache , D-cache , unified caches
4. Intel core i7 cache hierarchy
5. Writing Cache Friendly Code

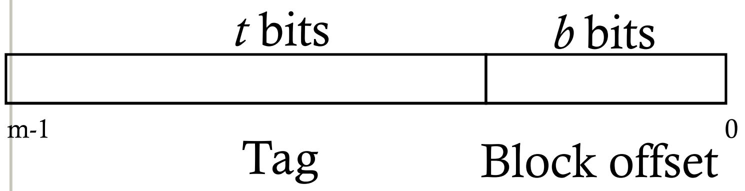
Fully Associative Cache Organization

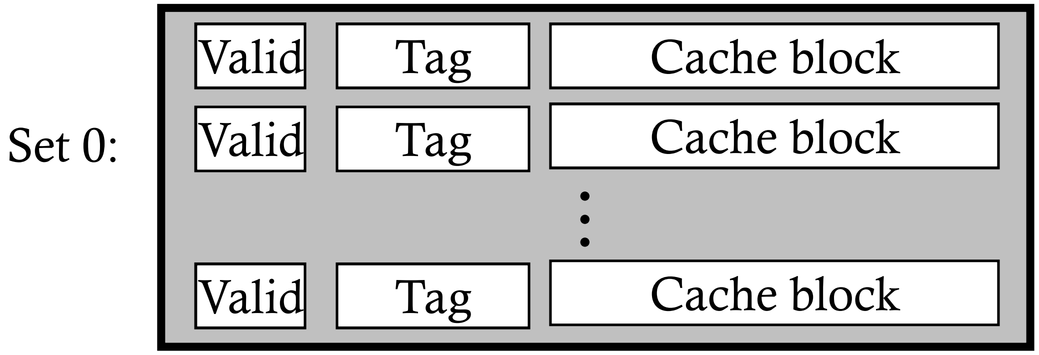


More circuitry and hence more expensive than Direct mapped and Set Associative Caches

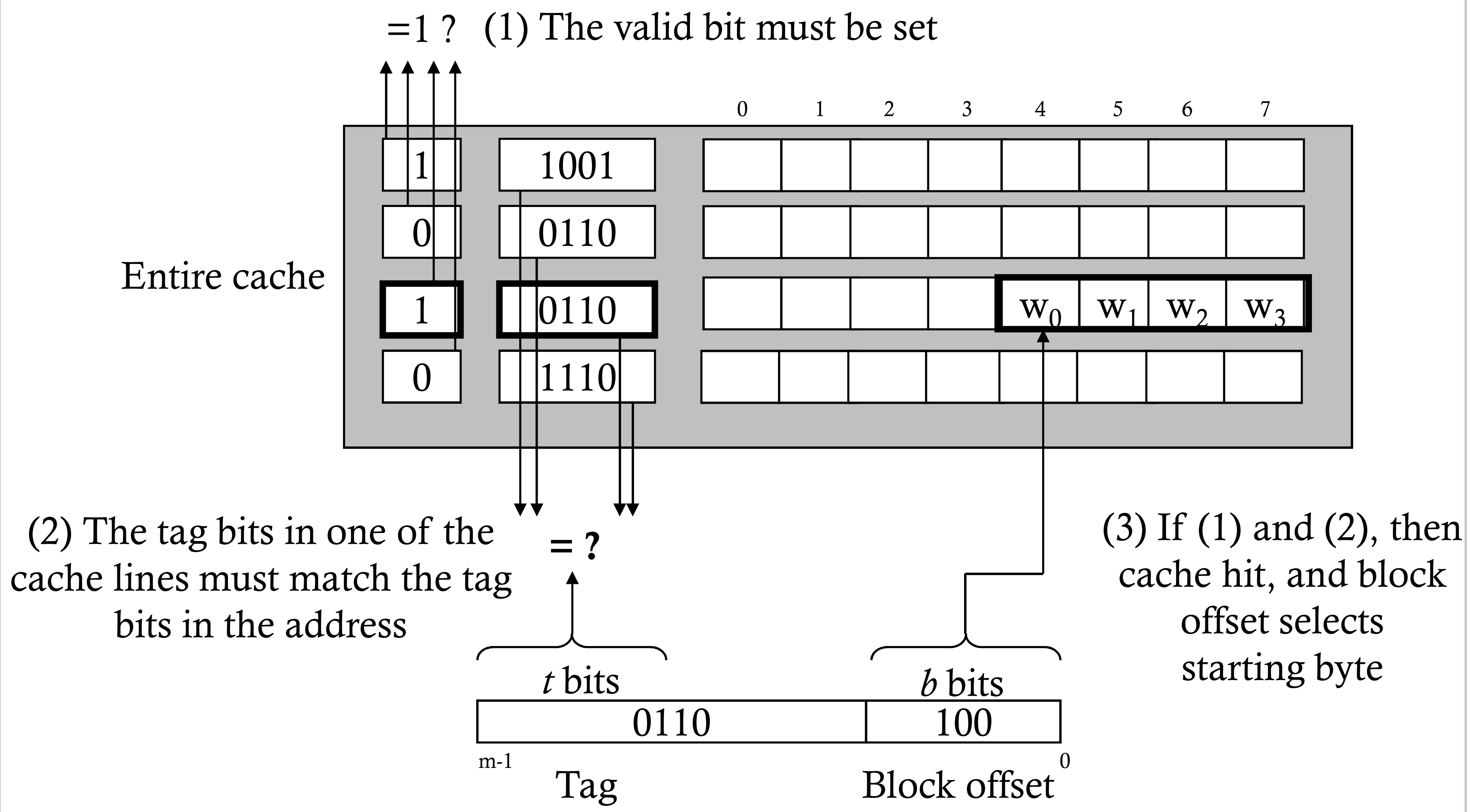
Looking up a memory address in Fully Associative Cache

The entire cache is one set, so by default set 0 is always selected

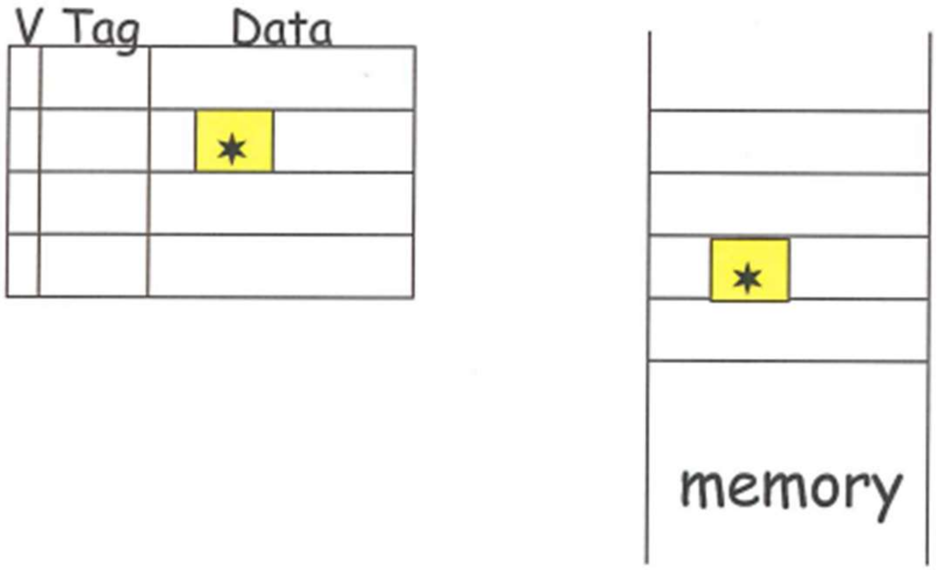




Looking up a memory address in Fully Associative Cache


=1 ?
(1) The valid bit must be set
0 1 2 3 4 5 6 7
Entire cache
1 1001
0 0110
1 0110
0 1110
w_0 w_1 w_2 w_3
(2) The tag bits in one of the cache lines must match the tag bits in the address
= ?
(3) If (1) and (2), then cache hit, and block offset selects starting byte
t bits
b bits
0110
100
m-1
0
Tag
Block offset

Implementing writes



1 *write through*

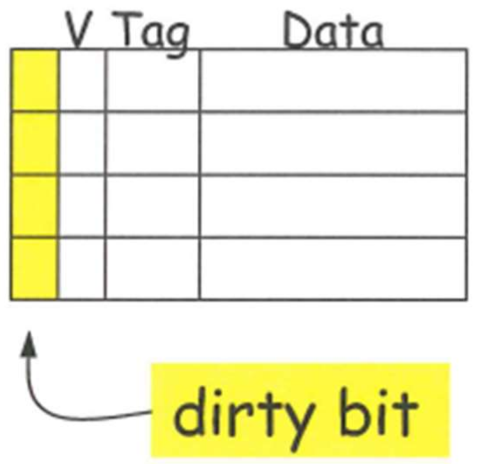
change data in the cache, and send the write to main memory

slow ☹ , but very little circuitry ☺

2 *write back*

- at first, change data in the cache
- write to memory only when necessary

dirty bit is set on a write, to identify blocks to be written back to memory



when a program completes, all dirty blocks must be written to memory. . .

2 *write back (continued)*

- faster ☺
  multiple stores to the same location result in only 1 main memory access
- more circuitry ☹
  - must maintain the dirty bit
  - *dirty miss* : a miss caused by a read or write to a block not in the cache, but the required block frame has its dirty bit set. So, there is a write of the dirty block, followed by a read of the requested block.

Writing during cache miss: (Two approaches)

- Write Alloc: Load block in cache and update word (often used along with Write back)
- Write No-Alloc (a.k.a.Write around): Just update memory (often used along with Write through)

How about 2 separate caches ?

V	Tag	Data

I-cache

- for instructions only
- can be rather small,

and still have excellent performance.

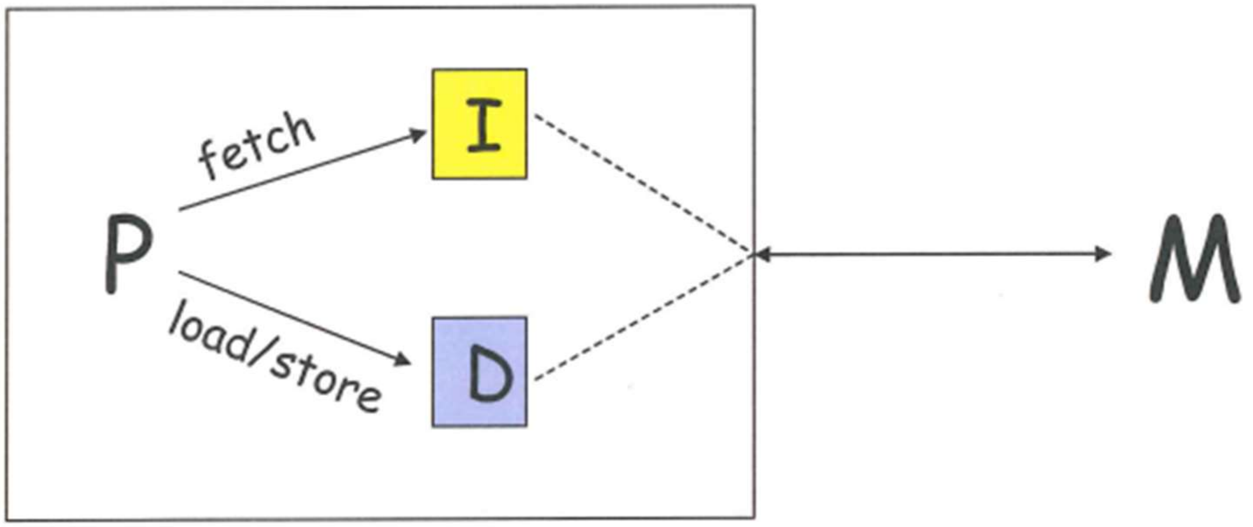
D-cache

- for data only
- needs to be fairly large

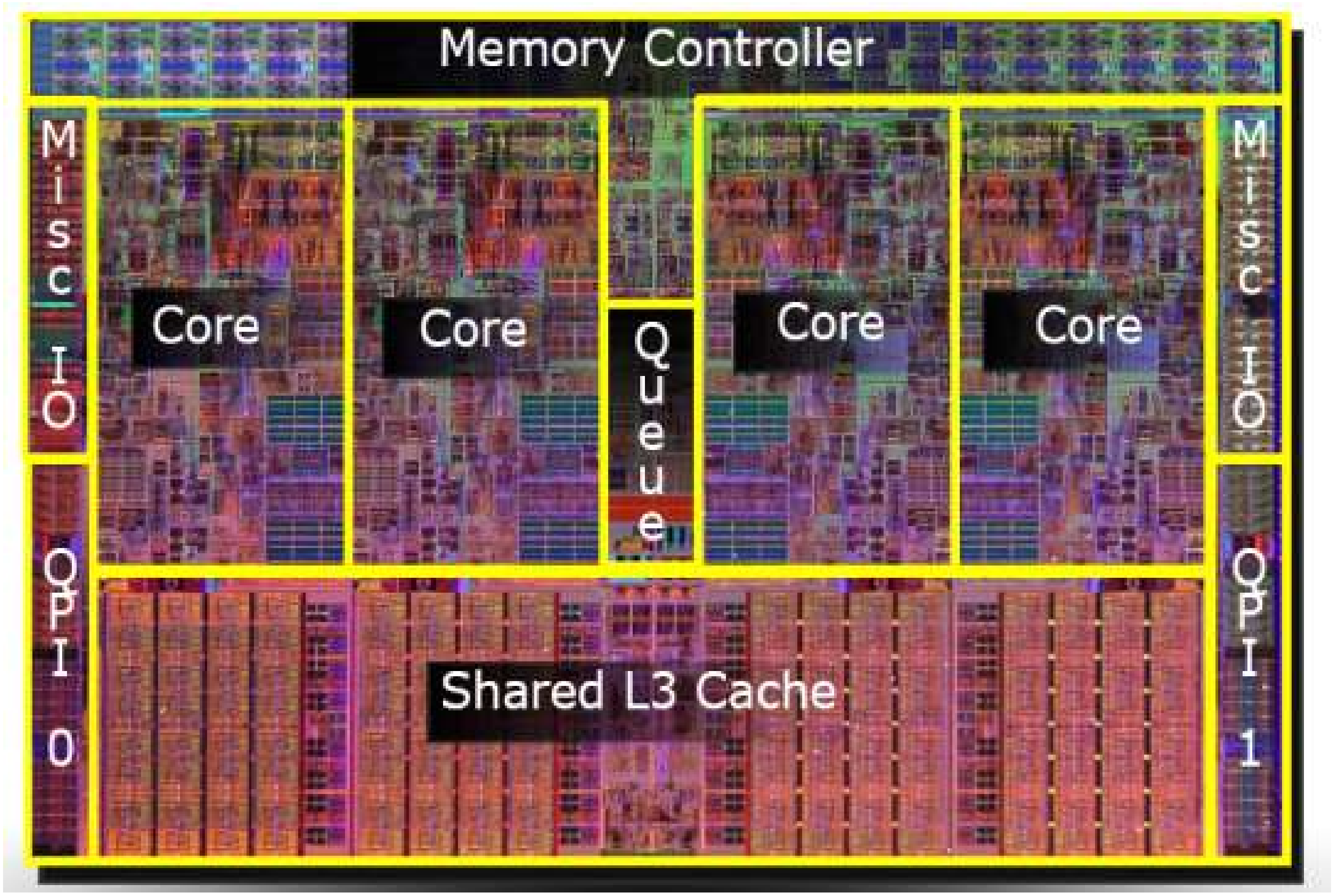
V	Tag	Data	V	Tag	Data

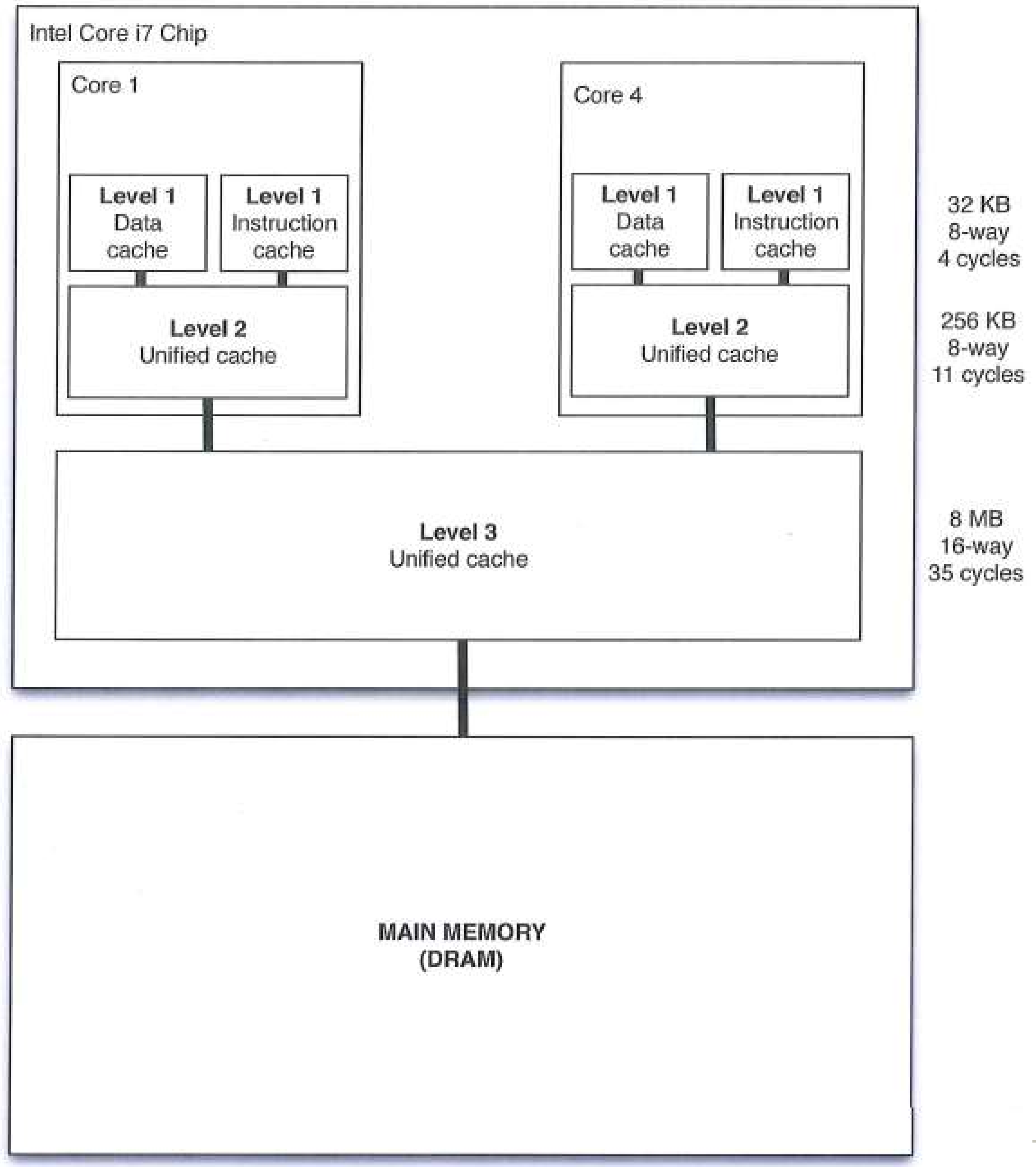
We can send memory accesses to the 2 caches independently. . .

☺ (increased parallelism)



Intel Nehalem Die Shot (Core i7 and later)



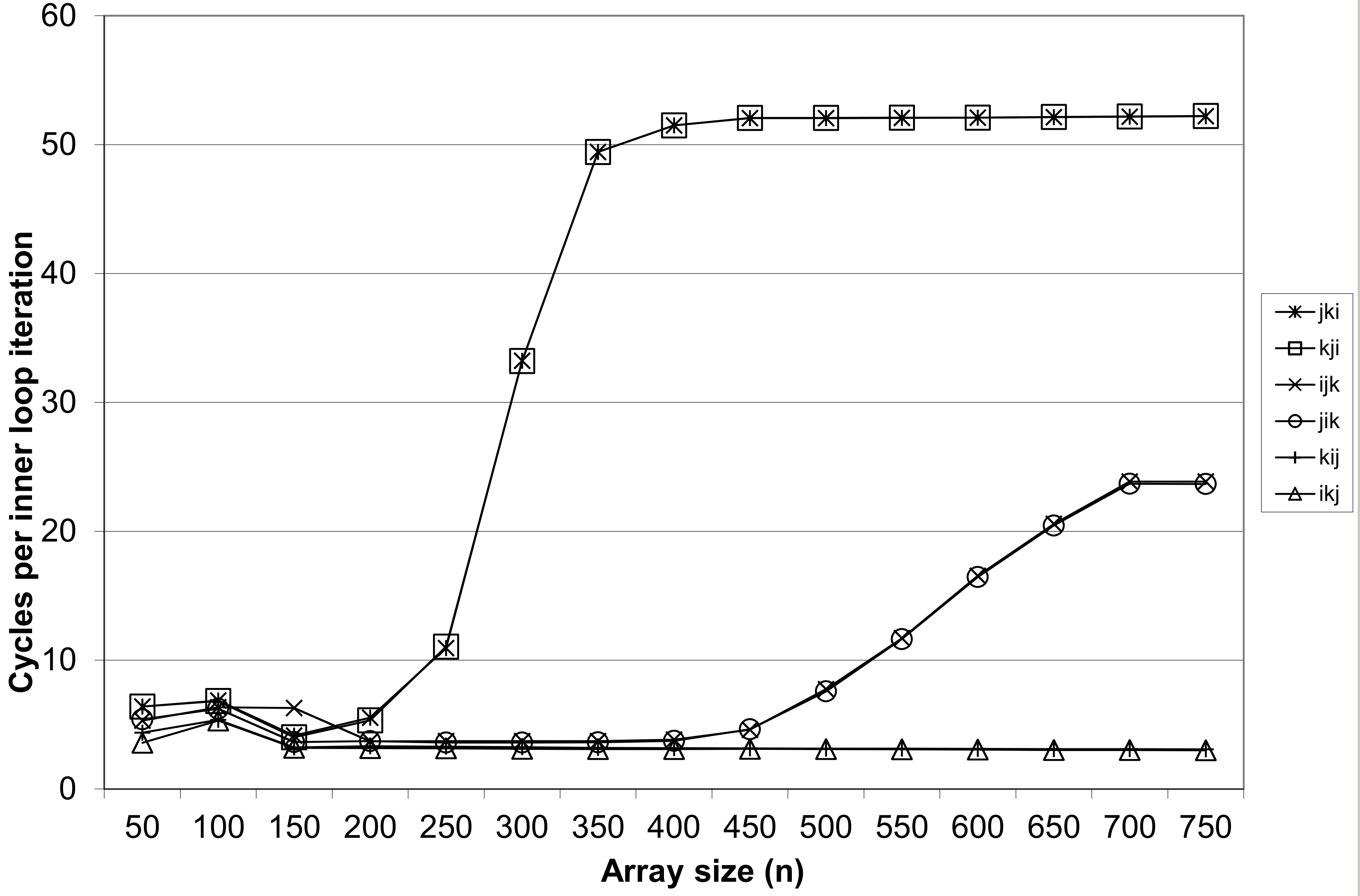

Intel Core i7 Chip
Core 1
Level 1 Data cache
Level 1 Instruction cache
Level 2 Unified cache
Core 4
Level 1 Data cache
Level 1 Instruction cache
Level 2 Unified cache
Level 3 Unified cache
MAIN MEMORY (DRAM)
32 KB 8-way 4 cycles
256 KB 8-way 11 cycles
8 MB 16-way 35 cycles

Matrix Multiply

Performs matrix multiplication using different loop combinations

For 1000 x 1000 double data type matrix multiplication on CSL machines

- Time taken for mmijk is : 6163814132 cycles or 1.71 seconds
- Time taken for mmjik is : 3349923284 cycles or 0.93 seconds
- Time taken for mmjki is : 9853809636 cycles or 2.74 seconds
- Time taken for mmkji is : 12881107088 cycles or 3.58 seconds
- Time taken for mmkij is : 2893624056 cycles or 0.80 seconds
- Time taken for mmikj is : 1721619796 cycles or 0.48 seconds



60
50
40
30
20
10
0
Cycles per inner loop iteration
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
Array size (n)
jki
kji
ijk
jik
kij
ikj

Writing Cache Friendly Code

1. Focus on the inner loops where bulk of computation and memory accesses occur

2. Maximize spatial locality by reading data objects sequentially with stride 1

3. Maximize temporal locality by reading a data object as often as possible once it has been read from memory.